(2)

Technical Report 1468
December 1991

Studies of Iterated Transform Image Compression and its Application to Color and DTED

R. D. Boss
E. W. Jacobs

ADMINISTRATIVE INFORMATION

This work was sponsored by the NOSC in-house Independent Exploratory Development Program (IED). The work was performed by members of Code 633, Naval Ocean Systems Center, San Diego, CA 92152-5000 during fiscal year 1991, funded under program element 0602936N.

Released by
J. C. Hicks, Head
Research Branch

Under authority of
R. H. Moore, Head
ASW Technology Division

MA

SUMMARY

OBJECTIVES

To present results for the application of various metrics and a new classification scheme to the iterated transform image compression technique. To present results for the compression of Digital Terrain Elevation Database (DTED) images and 24-bit per pixel color images, and to compare these results to other existing image compression techniques.

RESULTS AND CONCLUSIONS

Results show that a new classification technique based on *archetypes* can reduce encoding time while maintaining compression and fidelity. The use of various metrics has been determined to result in little differences in the final encoding quality thus indicating that the use of the root mean square (rms) metric is preferred (due to the computational simplicity associated with this metric).

Encodings of DTED data using an iterated transform algorithm modified to encode coastlines with increased accuracy are presented. These results compare favorably with an adaptive discrete cosine transformation (ADCT) and mean residual vector quantization algorithm. Results for two different iterated transform algorithms for encoding 24-bit per pixel (bpp) color images are presented. The first method transforms the red, green, blue (RGB) image to the luminance-chrominance representation before encoding, and results in performance similar to ADCT algorithms. The second method encodes the RGB subimages directly using common transformations, and does not perform as consistently as the luminance-chrominance method.

CONTENTS

CONTENTS (continued)

1. INTRODUCTION

The compression of images (and other 3-dimensional data arrays) is a field of rapidly growing importance. This report presents several results pertaining to this field, with particular emphasis on the application of a technique having its roots in the study of fractal objects and dynamical systems, namely iterated transformations. This report is divided into four main parts: (1) a brief introduction to the basic formalism and methodology of iterated transform image compression, (2) a section describing some results for alternative metrics and classification schemes pertaining to iterated transform image compression, (3) a section on the compression of the Digital Terrain Elevation Database (DTED), and (4) a section on the compression of 24-bit color images.

1.1 BACKGROUND

The iterated transform method has its roots in the theory of iterated function systems (IFS), which has been popularized and studied extensively by Barnsley (1988, Barnsley & Sloan, 1988). The method has received particular interest partly because of the fractal nature of the encodings. The iterated transform method was first presented and applied to gray scale images by Jacquin (1989, 1990*a* and 1990*b*). A thorough description of the iterated transform algorithm used here is given elsewhere (Fisher, Jacobs, & Boss, 1991 and Jacobs, Fisher, & Boss, 1991). The following is a brief overview of the technique.

Theoretical Background. The image is encoded in the form of an iterative system (a space and a map from the space to itself) $W : F \to F$. The space F is a complete metric space of images, and the mapping W (or some iterate of W) is a contraction. *The contractive mapping fixed point theorem ensures convergence to a fixed point upon iteration of W.* To compress an image, the goal is to construct the mapping W with fixed point "close" [based on a properly chosen metric $\delta(f, g)$] to a given image that is to be encoded, and such that W can be stored compactly. The collage theorem provides motivation that a good mapping can be found (Barnsley, 1988). Decoding then consists of iterating the mapping W from any initial image until the iterates converge to the fixed point.

Let $I = [0, 1]$ and I^n be the n-fold Cartesian product of I with itself. Let F be the space consisting of all graphs of real Lebesgue measurable functions $z = f(x, y)$ with

$(x, y, f(x,y)) \in I^3$. Let $D_1, \ldots, D_n$ and $R_1, \ldots, R_n$ be subsets of I^2 and $v_1, \ldots, v_n : I^3 \to I^3$ be some collection of maps. Define w_i as the restriction

$$w_i = v_i|_{D_i \times I}.$$

The maps $w_1, \ldots, w_n$ are said to *tile* I^2 if for all $f \in F$, $\bigcup_{i=1}^n w_i(f) \in F$. This means the following: for any image $f \in F$, each D_i defines a part of the image $f \cap (D_i \times I)$ to which w_i is restricted. When w_i is applied to this part, the result must be a graph of a function over R_i, and $I^2 = \cup_{i=1}^n R_i$. This means that the union $\cup_{i=1}^n w_i(f)$ yields a graph of a function over I^2 and that the R_i's are disjoint. The map W is defined as

$$W = \bigcup_{i=1}^{n} w_i. \tag{1}$$

The encoding procedure is as follows: Since the goal is to limit the memory required to specify W, I^2 is partitioned by geometrically simple sets R_i with $\cup_{i=1}^n R_i = I^2$. For each R_i, a $D_i \subset I^2$ and $w_i : D_i \times I \to I^3$ is sought such that $w_i(f)$ is as δ close to $f \cap (R_i \times I)$ as possible; that is,

$$\delta(f \cap (R_i \times I), w_i(f)) \tag{2}$$

is minimized. This is referred to as *covering* R_i with D_i. So that the contractive mapping fixed point theorem will ensure convergence to a fixed point, the w_i's must be chosen such that W or some iterated of W is contractive. The motivation for minimizing expression (2) is provided by the collage theorem.

Implementation. To limit the memory required to specify w_i, only maps of the form

$$w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \tag{3}$$

are considered, where w_i is restricted to $D_i \times I$. In the implementation described here, the D_i's and R_i's have been restricted to be squares, thereby further restricting the possible values for the coefficients a_i, b_i, c_i, d_i, e_i, and f_i in equation 3. Insisting that w_i map (the graph above) D_i to (a graph above) R_i while minimizing condition (2), determines s_i and o_i. (If any point in the transformed graph over R_i is outside the allowed range of pixel values, it is clipped.) In this way, w_i is determined uniquely for a chosen metric. The results in sections 2 and 3 use the root mean square error (δ_{rms}) as the chosen metric. Further discussion and some results for other metrics are discussed in section 1.2.

Let **R** be the collection of subsets of I^2 from which the R_i are chosen, and let **D** be the collection of subsets of I^2 from which the D_i are chosen. In the implementations described here, **R** and **D** were chosen to be squares of varying sizes. Although the D_i's and R_i's are not strictly the domains and ranges of the w_i's, the terminology will be used because it is descriptive.

For a given choice of **R** and **D**, the encoding problem is reduced to choosing a set $\{R_i\} \subset \mathbf{R}$, and the corresponding set $\{D_i\} \subset \mathbf{D}$, such that good compression and an accurate encoding of the image results. The method used to find good D_i's determines how much computation time the encoding takes. For each R_i, a search for D_i through all of **D** would clearly result in the choice that would best minimize expression (2), but for applications for which encoding time is a consideration, such a search may require too much computation time. Therefore, a classification scheme can be used in the following way. First, all the domains are classified into a set of classes. During encoding, each range square is classified using the same classification procedure as the domains, and a search for the optimal domain square in the same class (or similar classes) is performed. If the classification is fast, and separates the domains and ranges such that they cover well [i.e., expression (2) is effectively minimized], then encoding time will be reduced with little loss in image fidelity. The scheme employed for the encodings presented in sections 2 and 3 was relatively straightforward. Seventy-two classes were defined by the brightness and edge-like character of the quadrants of each square. The classification scheme began by computing the average intensity for each quadrant of the (range or domain) square. Then a symmetry operation was applied to force the square into an orientation with its brightest quadrant in the upper left quadrant, and to put the second brightest quadrant into the upper right quadrant (or the third brightest if the second brightest could not be so oriented). This process divided the squares into three main classes (and defined a symmetry operation for each square). Each of these three main classes was then subdivided by determining the amount of the variation of the average brightness of the subquadrants for each quadrant of the square. The quadrants of the square were thereby ordered from first to fourth by the amount of variation within each quadrant. There are 24 possible permutations for the order of the relative brightness variations, resulting in a total number of 72 classes. In section 1.3, an alternative classification scheme using *archetypes* is examined.

In the implementation described by Fisher, et al. (1991) and Jacobs, et al. (1991), which was used to encode the images in sections 2 and 3, a recursive quadtree partitioning method was used to reduce the error in regions of high variability and to increase compression in "flat" areas of the image. The quadtree partitioning allowed the range squares to vary in size. For a given R_i, if a w_i could not be found such that expression (2) was less than a predetermined error criteria (e_c), the range square was subdivided into four equal squares, and the process was repeated until the error criteria was satisfied or a range square of a predetermined minimum size r_{min} was reached. For range squares of size r_{min}, the best w was stored whether or not e_c was satisfied.

1.2 METRICS

Each R_i of $\mathbf{R}$, and D_i of $\mathbf{D}$ are of the form

$$R_i = \begin{bmatrix} r_{i_{11}} & r_{i_{12}} & \dots & r_{i_{1m}} \\ r_{i_{21}} & r_{i_{22}} & \dots & r_{i_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ r_{i_{m1}} & r_{i_{m2}} & \dots & r_{i_{mm}} \end{bmatrix} \quad D_i = \begin{bmatrix} d_{i_{11}} & d_{i_{12}} & \dots & d_{i_{1m}} \\ d_{i_{21}} & d_{i_{22}} & \dots & d_{i_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ d_{i_{m1}} & d_{i_{m2}} & \dots & d_{i_{mm}} \end{bmatrix}.$$

The transformed domain (which covers the range) is of the form

$$w_i(D_i) = \begin{bmatrix} s_i * d_{i_{11}} + o_i & s_i * d_{i_{12}} + o_i & \dots & s_i * d_{i_{1m}} + o_i \\ s_i * d_{i_{21}} + o_i & s_i * d_{i_{22}} + o_i & \dots & s_i * d_{i_{2m}} + o_i \\ \vdots & \vdots & \ddots & \vdots \\ s_i * d_{i_{m1}} + o_i & s_i * d_{i_{m2}} + o_i & \dots & s_i * d_{i_{mm}} + o_i \end{bmatrix}. \tag{4}$$

The distance between the transformed domain and the range is measured by some metric, such as one of the L^n metrics. These metrics can be easily computed since the distance is given by

$$L^n = \delta_n(R_i, w_i(D_i)) = \frac{\left\{\sum_{j=1}^{m}\sum_{k=1}^{m}\left[\left(s_i * d_{i_{jk}} + o_i\right) - r_{i_{jk}}\right]^n\right\}^{\frac{1}{n}}}{m}$$

The primary task during the encoding process is minimizing the error [the distance between R_i and $w_i(D_i)$]. Not surprisingly, the final result of the encoding is dependent upon the choice of the metric used during the minimization.

There exists in the literature some debate about which metric should be used in minimizing the error of an encoding to achieve the most visually appealing image. In the case of iterated transforms, the choice of the metric is also of interest since the convergence of mappings is dependent on the metric used. To date, only the L^∞ and

L^2 metrics have been employed in iterated transform algorithms. In the following paragraphs, we briefly discuss some results using other metrics.

The results for encoding several images using the L^n, $n \in \{1,2,3,4,5,\infty\}$ indicated two primary conclusions. First, as might be expected, when the error of the resultant encoding was measured in a given metric, the encoding performed by minimizing that metric had the least error. Second, for the metrics tested, the differences between the encodings were very small, i.e., it was difficult to visually differentiate between the encoded images, and there was no consensus as to which metric resulted in the best appearing images.

Since there is no clear choice of metrics based on the visual quality of the encoded images, other factors can be considered in choosing a metric, namely, ease of implementation. The task of determining what values of s and o minimize condition (2) is computationally simple only for the L^2 metric. Since this gain in speed and simplicity more than outweighs any small perceived visual gain obtained by using another metric, results in this report use L^2 (δ_{rms}) during encoding, and all results for the error of encodings are also measured in δ_{rms}.

1.3 CLASSIFICATION METHODS

The main point of this section is to introduce a new classification method, based on *archetypes*, which can be used to increase the speed of iterated transform compression alogrithms. Finding a set of archetypes uses many of the same techniques used in making a vector codebook for a vector quantization (VQ) encoder. Vector quantization is a well-studied technique commonly used to compress digital images. Because it introduces some of basic ideas that will be used to construct archetypes, the following section describes, in some detail, a method used to construct a vector codebook. The specific method discussed was first decribed by Linde, Buzo, and Gray, (1980). The following discussion will serve to better understand the similarities and differences between archetypes and VQ codebooks, and also to better understand the method used to produce the mean residual VQ encodings in section 2 of this report.

Vector Quantization Codebooks. In VQ, an image (a large 2-dimensional array of pixels) is subdivided into many smaller arrays, called the *target vectors*. Each target vector is encoded as a *code vector*, which is drawn from a list of code vectors, called the *codebook*. The storage of the image is then simply a list of the codes that indicates which code vector in the codebook best matches the sequence of target vectors that make up the image. The subsequent reconstruction of the image is then simply a pasting together of the indicated code vectors.

Encoding an image is a simple procedure. For each target vector, a search through the code vectors is performed to find the one that (measured in a given metric) is closest. Therefore, the most interesting problem in such an encoding scheme, and the problem that has been the focus of most of the research in this field, is the construction of the codebook.

The ideal codebook would be that set of vectors for which the total error of the reconstruction is zero. To have a codebook for which this is possible (for any image) would require a code vector which has as many bits as the original target vector, i.e., there would be no compression. A useful codebook would be one for which it is possible to "accurately" encode any image with only a modest amount of information. This requires that the codebook be composed of a limited number of code vectors (so that good compression is achieved), and which have a wide variety of characteristics (so that images can be encoded accurately). To simplify the codebook, a common technique is to subtract the mean value of each target vector from each target vector, and include it separately as part of the encoded image. Therefore, only variations about the mean need to be contained in the codebook.

A specific method to generate a codebook based on the general method of Linde et al. (1980) is shown in figure 1. The initial state is assumed to be a single code vector (of all zeros) and a set of "teaching" vectors (all of mean zero). A search is then performed to determine those two elements of the teaching set which, when used to encode the entire set, gives minimum square error. The search is *not* performed for all possible sets of any two elements because such a search is too time-consuming. For example, a full search for the first step, for a teaching set of 8×8 vectors drawn from five 512×512 images requires the computation of of 20,478 square errors for each of 209,725,440 different combinations. Consequently, the two elements are determined from a search of a randomly chosen subset of 200 elements. Each of the 19,900 possible combinations is checked, with the minimum error pair beginning used as the initial code vectors.

The entire set of teaching vectors is then sorted into subsets by determining which of the new code vectors best encodes them. Each subset is then used to compute a new best code vector. In this case, the best code vector is simply determined by computing the average of all the vectors in the subset. This process is then repeated until a self-consistent set of code vectors is obtained.

If the number of self-consistent code vectors is less than the desired number of code vectors, then each code vector is bifurcated into two new code vectors, by dividing the subset of teaching vectors, precisely as described previously. This process of iterating to self-consistency then bifurcating is continued until the number of desired code vectors is obtained. (Note, the only problem encountered with this process is when a subset of the teaching vectors consists of exactly one vector, in which case the subset cannot be divided. To continue the process of bifurcation for this case, after all other subsets have been divided, the subset that has the most elements is divided once more. This extra division of the largest subset is performed for all subsets of size one, then, and only then, is the self-consistency iteration performed.)

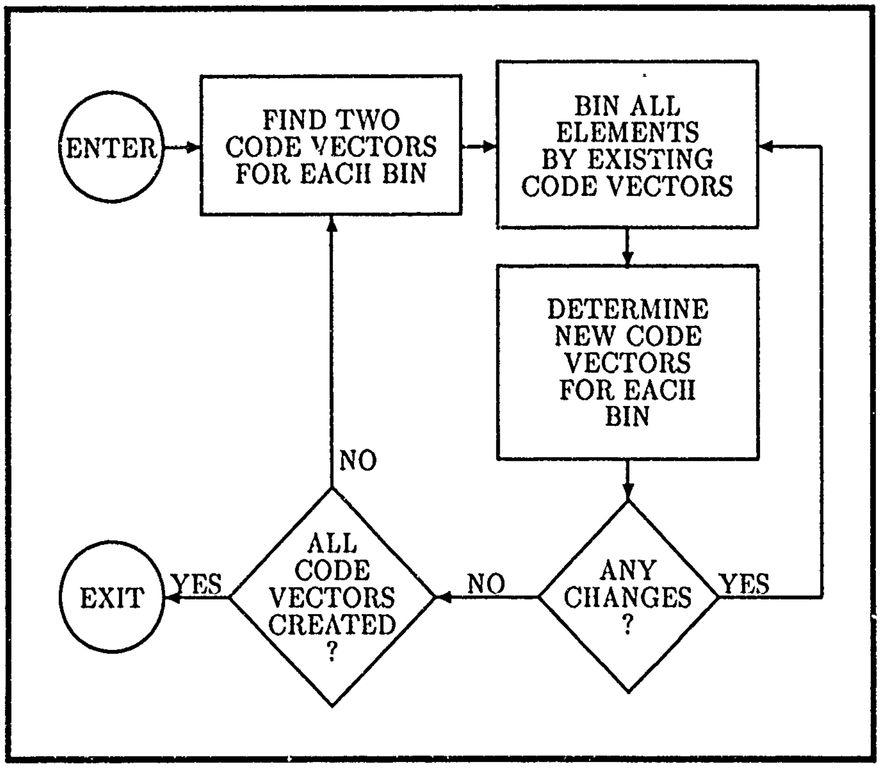
ENTER
FIND TWO CODE VECTORS FOR EACH BIN
BIN ALL ELEMENTS BY EXISTING CODE VECTORS
DETERMINE NEW CODE VECTORS FOR EACH BIN
ANY CHANGES ?
YES
NO
ALL CODE VECTORS CREATED ?
NO
YES
EXIT

Figure 1 . A flowchart showing how to make a codebook.

Archetype Classification Applied to Iterated Transforms. In an iterated transform algorithm, the purpose of classification is (as mentioned previously) to reduce the number of domains that has to be checked to find an acceptable covering. Ideally, it would be possible to compute some simple property (or set of properties) that would result in identifying the optimal covering with no need for an actual computation of the error. The classification mentioned at the end of section 1 is based upon the idea that by orienting blocks in a canonical form (based upon brightness), and then subdividing these primary classes further by the location of strong edges, it should be possible to find good coverings with minimal computation. In fact, this classification method performs moderately well. However, in a few cases the optimal covering can only be found by searching through all classes, i.e., the classification procedure fails to classify "similar" objects into nearby classes. Consequently, it seems likely that some other classification scheme may perform better.

Since the goal of the encoding scheme is to find good coverings, it is reasonable to try to base the classification scheme on finding good coverings. This is what the *archetype* classification scheme is designed to do. So, what is an archetype? An archetype is that member of a class that is best able to cover all the other members of the class. The following describes how archetypes are determined and used.

The archetype, $\mathcal{A}$, for a set of vectors is determined by searching through the entire set to find that member of the set which can *cover* the other members best. That is, for each member, $\mathcal{E}_l$, $l \in \{1,2,\ldots,n_{elements}\}$, the best transform, w, of the form of equation (3) is found for every other member, $\mathcal{E}_k$, $k \in \{1,2,\ldots,n_{elements}\}$ and $k \neq l$, such that the rms distance, $\delta_{rms_{lk}}$, given by

$$\delta_{rms_{lk}} = \frac{\left[\sum_{i=1}^{n}\sum_{j=1}^{n}(w(\mathcal{E}_l) - \mathcal{E}_k)^2\right]^{\frac{1}{2}}}{n}$$

is minimized. The archetype is then chosen as that member for which the total error, Δ_{rms_l}, which is given by

$$\Delta_{rms_l} = \sum_{k=1}^{n} \delta_{rms_{lk}}$$

is minimum. That is $\mathcal{A} = \mathcal{E}_l$ if and only if $\Delta_{rms_l} \leq \Delta_{rms_k}$ for all k.

If some method exists to separate the entire set of vectors into classes, then each of the classes (i.e., each of the sets) has a definable archetype. Clearly, the determination of the archetypes for some set of classes is given by the flowchart in figure 2a.

A more useful set of archetypes can be generated if, rather than stopping once the archetypes are determined, the archetypes are then used to perform the sorting of the vectors into classes (i.e., the archetype that best covers a given vector defines the class into which the vector is classified), with the reclassified vectors then being used to determine new archetypes. This process can then be iterated to self-consistency. A flowchart of this process is shown in figure 2b.

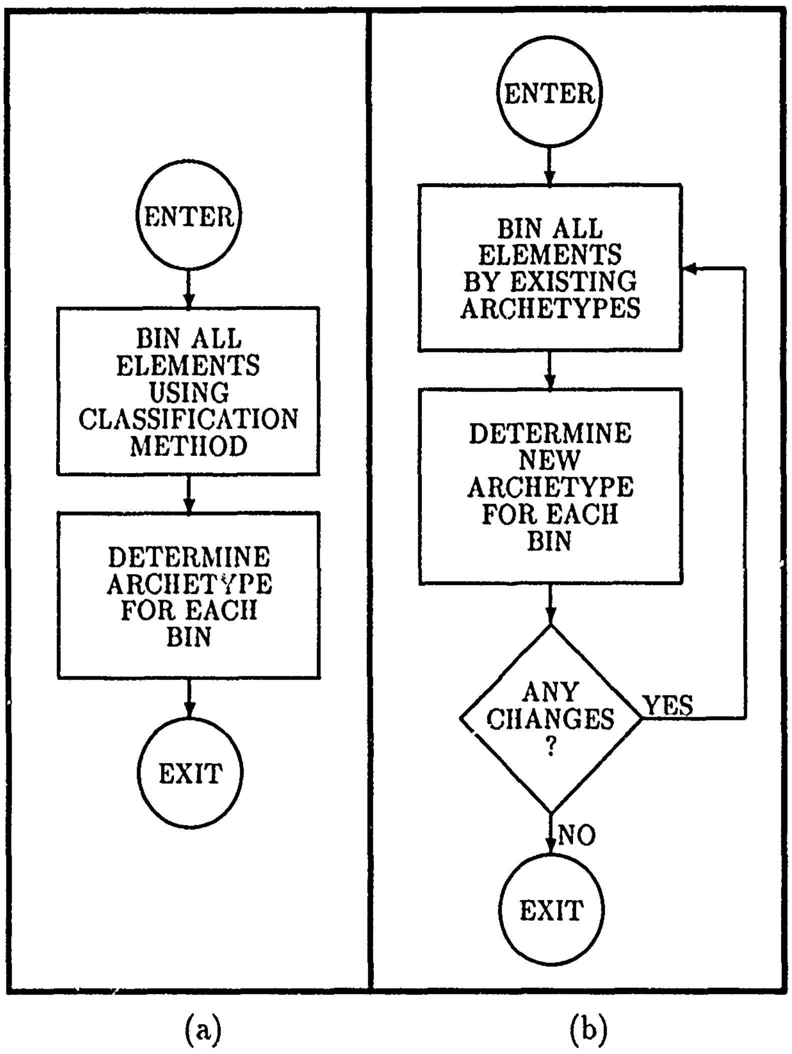
ENTER
BIN ALL ELEMENTS USING CLASSIFICATION METHOD
DETERMINE ARCHETYPE FOR EACH BIN
EXIT
(a)
ENTER
BIN ALL ELEMENTS BY EXISTING ARCHETYPES
DETERMINE NEW ARCHETYPE FOR EACH BIN
ANY CHANGES ?
YES
NO
EXIT
(b)

Figure 2 . A flowchart showing how to make self-consistent archetypes.

The determination of archetypes is obviously very similar to the determination of a vector quantization codebook; however, there are major differences. The most fundamental difference is, for archetyping the goal is not to find out how much two vectors look like each other, but rather to find a way of deciding how much two vectors can be made (by means of the transformation w) to look like each other. In additon, the code vectors in VQ algorithms are computed from the set of teaching vectors, but are not actually a member of the set of teaching vectors. This is different from an archetype, which is a member of the teaching set.

So, how would one use archetypes in an iterated transform algorithm? One method would be to determine a set of archetypes for an image, and then encode the image using the set of archetypes as the set D. This idea borrows the basic concept of VQ, in which a small set of appropriate code vectors make up a codebook. The set **D** will have very few, but carefully chosen members. As a result, a large number of relatively compactly specified w_i's can be used to encode an image. This method is the opposite extreme of an algorithm where the set **D** is large, the w_i's require more bits for storage, and fewer w_i's are used to encode the image. This method of using the computed set of archtypes for the set **D** actually produces good encodings based on fidelity versus compression performance. The problem with this method is that determining the archetypes entails performing almost the same calculations necessary in encoding the image with a more general set **D**. Therefore, even though **D** is small in number, the encoding time is not reduced.

A second method for employing archetypes in an iterated transform algorithm is to use archetypes as a classification method. Following the procedure outlined above, a set of archetypes can be found from an image or set of images. This set of archetypes can then be used to define the classes into which ranges and domains are binned at the start of the iterated transform encoding procedure, i.e., each range and domain is assigned to the class of the archetype that covers it best. Once again, this idea borrows a technique used in VQ. In VQ, the codebook is generated from a set of images which does not typically include the images the algorithm will ultimately be used to encode. With this classification method, the classes are defined from images other than the ones that will ultimately be encoded. If the images used to find the archetypes are sufficiently general, then a good set of archetypes to classify domains and ranges for a broad variety of images will result. If for a specific application, the set of images to be encoded have similar characteristics, archetypes could be found from a set of images with these same characteristics, and the classification method might work particularly well. Before examining some results, it should be noted that the idea of using predetermined archetypes (from images other that the one being encoded) is somewhat contrary to the basic concept of the iterated transform method. Therefore, it is necessary to emphasize that the archetypes are being used only to classify domains and ranges in an efficient way so as to reduce encoding time, and have no other role in the encoding procedure.

Figures 3 and 4 show data for an algorithm encoding 256×256 Lena using both the standard classification method and an archetype classification method. These encoding algorithms used only 16×16 domains covering 8×8 ranges. The encodings in this section were at approximately 16:1 compression. The Δ symbols show the peak signal-to-noise ratio (PSNR) versus number of bins searched for a set of runs using

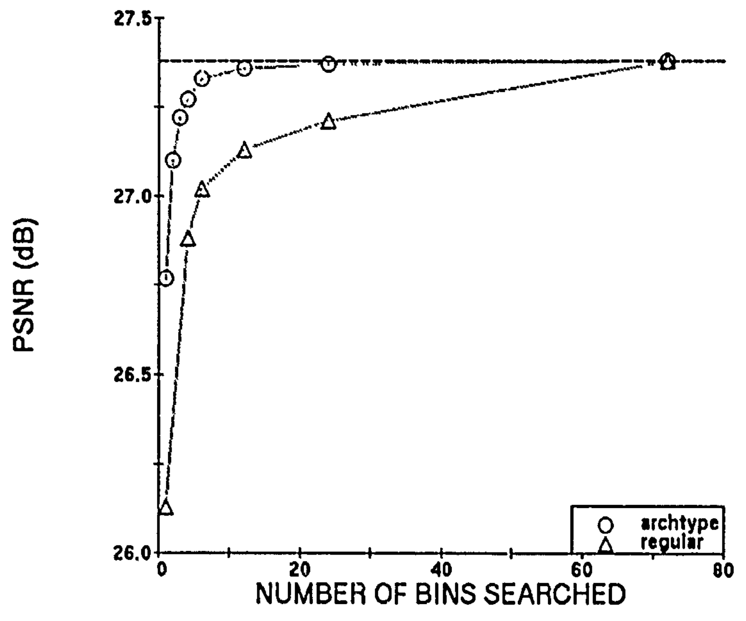

Figure 3 . PSNR vs. number of bins searched.

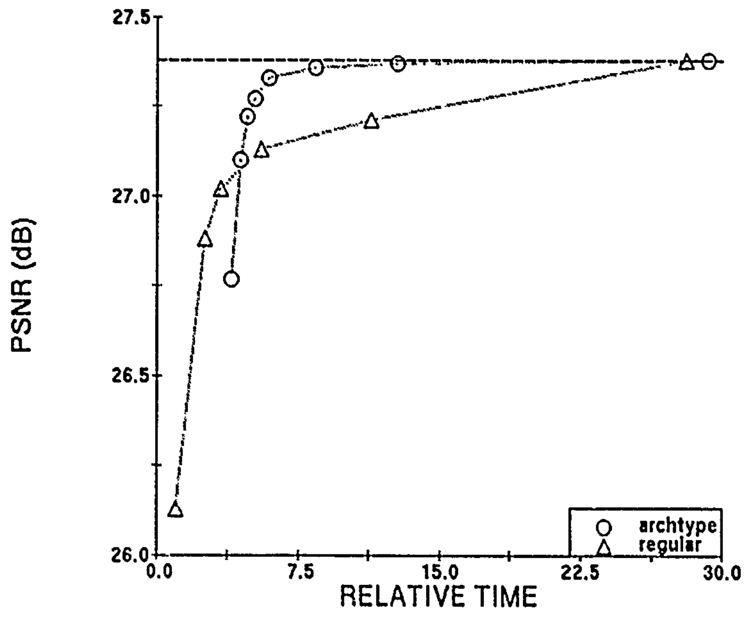

Figure 4 . PSNR vs. relative time.

the standard classification method. PSNR is defined as,

$$PSNR = -20 * \log_{10}\left(\frac{rms}{2^8 - 1}\right),$$

where rms is the root mean square error of the reconstructed image. Data are shown for 1, 4, 6, 12, 24, and 72 classes searched. The accuracy of the encoding increases as the number of classes searched increases, as does the time necessary to perform the search, where the complete 72-class search yields the best possible result for this encoding. The $\circ$ symbols show the results of the search using an archetype-based classification method. A set of 72 archetypes found from five images (a dog image, two images of people, one image of a city, and an image of an airfield) were used to define the classes. The Lena image was not included in this set of images. Data are shown for searches of 1, 2, 3, 4, 6, 12, 24, and 72 of the 72 classes. Figure 3 shows that, for a given number of classes searched, the archetype classification scheme results in an improvement in the accuracy of the encoded image. This demonstrates that the archetype classification method does indeed separate the vectors into classes that are better able to cover each other. In figure 4, note that, for a given number of classes searched, the total relative encoding time for the archetype classification encodings are longer than the standard encodings. This is because sorting the domains and ranges in the archetype classification scheme requires more time than it does for the standard classification method. Therefore, to outperform the standard classification method, the archetype method must yield more accurate encodings while searching through fewer classes. Indeed, figure 4 shows that the curves for the archetype classification and standard classification cross over, thus indicating that the archetype classification can (by searching fewer classes) result in improvements in both encoding accuracy and encoding speed.

2. APPLICATION TO DTED

Application of image compression to geographic map data is of particular interest to the Navy. Geographic map data comes in a variety of formats, and there has been extensive work done in compressing map databases for various applications. In this section, the problem of compression of the Digital Terrain Elevation Database (DTED) is addressed. The database consists of elevation data for a grid of longitude–latitude coordinates where the grid points are roughly 100 meters apart. A complete description of DTED can be obtained from the Defence Mapping Agency. Alward and Nicholls (1986) examined hierarchical data structures as applied to DTED. The data structures result in some data compression, although compression was not the primary goal of that investigation. In this section, the problem of interest is evaluating the performance of the iterated transform method on DTED. This would be useful for applications where data compression is the primary concern, and issues like hierarchical structure, access time, and decoding time are of secondary importance. As a means of evaluation, the DTED images were also compressed with an adaptive discrete cosine transform algorithm (ADCT) and mean residual vector quantization algorithm (MRVQ). DTED data can be thought of in terms of a gray scale image where the longitude and latitude identify the pixel, and the elevation is the pixel value. For the purpose of possible Navy application, the data were transformed from their original linear scale between 0 and 10000 meters, to a logarithmic scale between 0 and 255. The quantization results in a compression from 14 bits per datapoint to 8 bits per datapoint. This logarithmic scale, shown in figure 5, represents lower elevations more accurately than higher elevations, the rationale for this being that lower elevations areas are more likely to be important for Navy applications, and that nearly all of the earth's surface (particularly near coastlines) is at relatively low elevation.

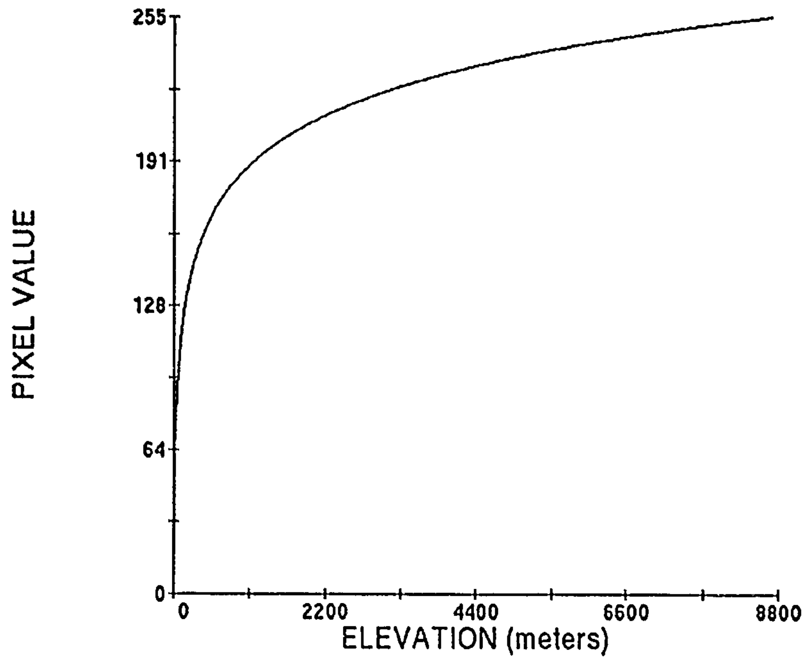

Figure 5 . The scale transformation used for the DTED data.

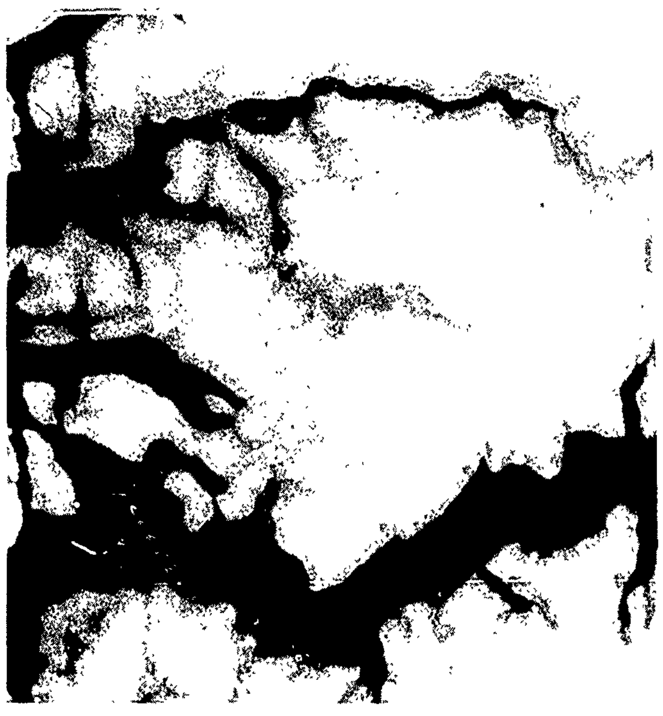

Figure 6 . The original DTED Norway image.

Tests in this section were performed on a 512 × 512 section located in the fiords of Norway. This test image is shown in figure 6. This section of Norway covers the section of the earth from 5° to 5° 25.6' east and 61° to 61° 12.8' north. In figure 13, the same data are shown in false color. (note: all color images are collected at the end of the report.) The elevation range for each color band is shown in figure 13. This section of DTED was chosen because the topology of the fiords serve as a severe test of the fidelity of the compression methods. A more thorough study where a broad area of the database containing a representative amount of flat and mountainous regions would be necessary in determining the compression and fidelity possible for the complete database.

The iterated transform algorithm was identical to that used by Jacobs et al. (1991) except for one significant modification. The algorithm was modified to encode sections of the image on coastlines with increased accuracy. For image sections that contained coastline, the error criteria was tightened. This resulted in more segmentation, and therefore higher fidelity in these areas. The ADCT algorithm used was similar to that described by Chen and Pratt (1984), except for a modification similar to that described above for the iterated transform algorithm. To encode coastlines more accurately, the decision level quantizer table was compressed for sections of the image on coastlines. Improving the fidelity at the coastlines resulted in a decrease in the overall compression-fidelity performance . The MRVQ algorithm used was as described in section 1.3. A more complete description of the method has been described by Linde et al. (1980). Codebooks were generated from two different training sets. One training set included 3 sections of DTED similar to (but not including) the section tested. The other codebook was generated from a diverse set of five digitized photographs, all distinctl; different from DTED images (a collection of faces, aerial photos, etc.).

The compression methods being considered are properly applied to images with a dynamic range appropriate to the number of bits used to store the image, and a relatively smooth distribution of gray level intensity values over this range. In figure 7, the number of datapoints at each (nonzero) elevation is plotted versus elevation for the 512 $\times$ 512 section of DTED, which is 1° east of that section shown in figure 6. In figure 8, this section is displayed as a series of "contours," sea level is displayed in gray, and all pixels with an elevation that is a multiple of 100 ($\pm$ 2) are displayed in black, with all other elevations being displayed in white. It appears that the majority of the data contained in this section of DTED were created from 100-meter contour maps, the result being that a disproportionate number of datapoints are at multiples of 100 meters. In the southeast corner of the map, it can be seen that the number of points at 100 meters are far more dispersed. In this section of the map, roughly the number of datapoints that would be expected based on random elevations are present. In figure 7, it is also evident that between the 100-meter multiples, quantization is also present on a finer level. The biased quantization illustrated by figures 7 and 8 is evident in other sections of DTED, including the section of figure 6.

Because of this biased quantization, the reconstructed image resulting from compressed encodings will have a smoother distribution of pixel values than the original image. This will lead to an artificially poorer measured fidelity of the encoded images. In areas of DTED where the data are "properly" digitized, this situation will not occur. Because not all the data are quantized at a course resolution, but only most of it, taking advantage of this quantization is not simple. Although it has not been done here, before applying lossy compression techniques such as iterated transforms, ADCT, or VQ, requantization of the data in such a way that the majority of the datapoints retain their correct values could result overall improved performance.

Figures 9, 10, and 11 show gray scale reconstructed images from typical encodings of the image in figure 6 using iterated transforms, ADCT, and MRVQ respectively.

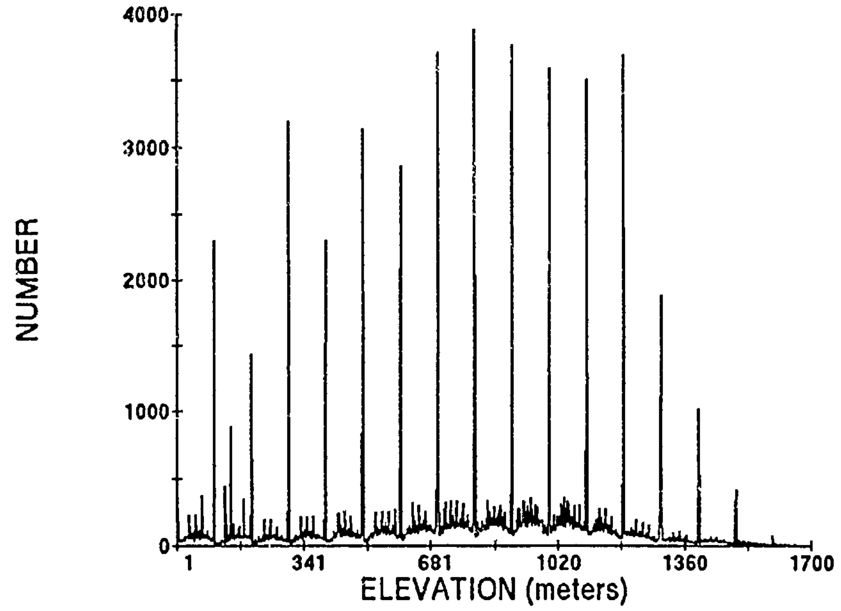

Figure 7 . The distribution of elevations for a section of DTED.

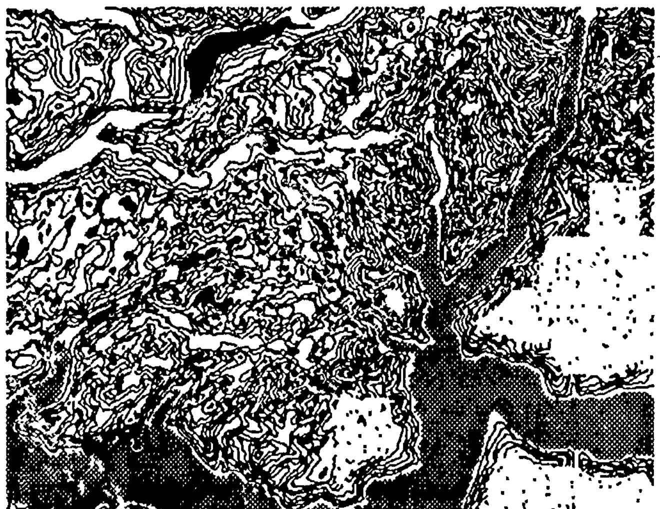

Figure 8 . Pixels of value 100 ±2 meters.

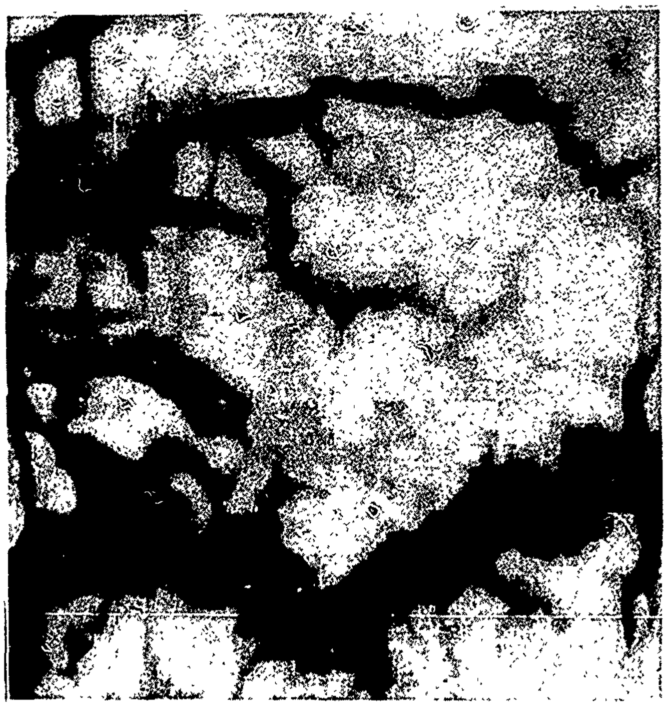

Figure 9 . The decoded iterated transform Norway image, compression = 44.86:1, PSNR = 32.98 dB.

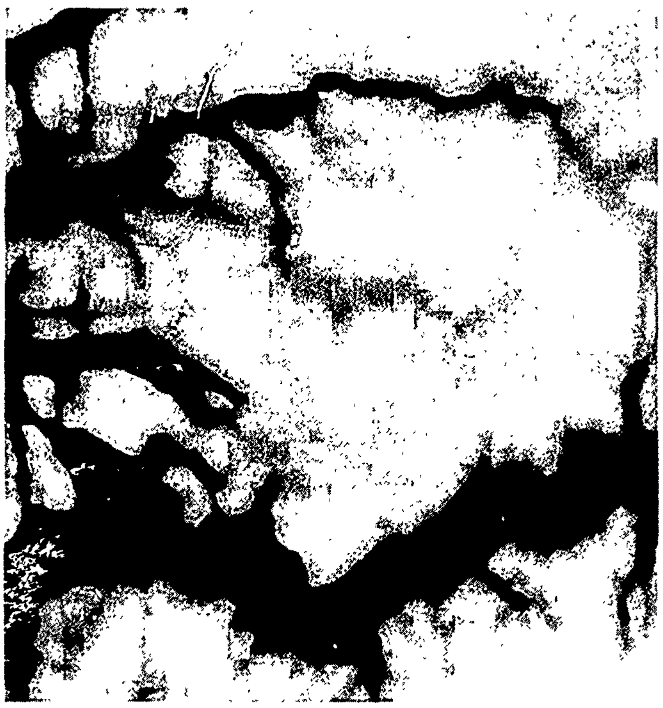

Figure 10 . The decoded ADCT Norway image, compression = 30.30:1, PSNR = 32.52 dB.

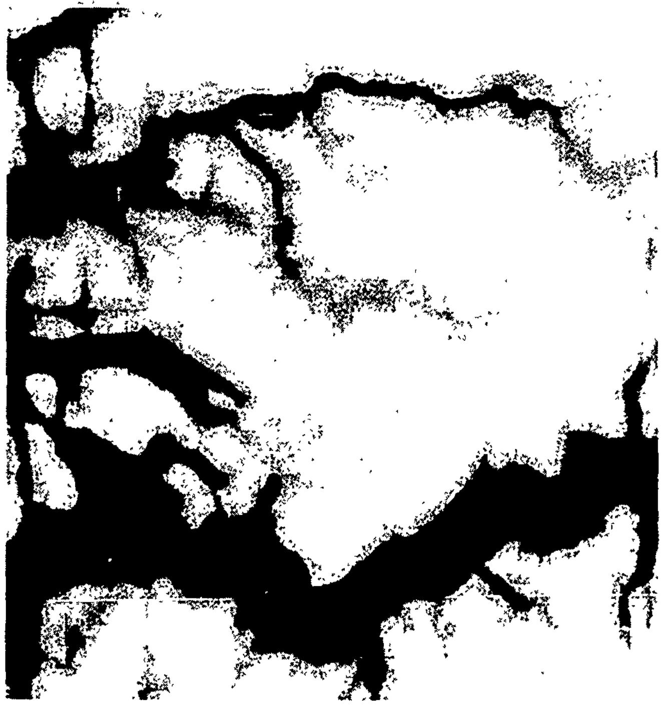
Figure 11 . The decoded MRVQ Norway image, compression = 32:1, PSNR = 31.16 dB.

Table 1 . Compression and fidelity results for encodings of DTED.

Method	Compression	PSNR(dB)
Iterated Transforms	44.86:1	32.98
Iterated Transforms	21.49:1	35.08
ADCT	30.30:1	32.52
ADCT	21.08:1	34.92
MRVQ (general codebook)	32.00:1	31.16
MRVQ (DTED codebook)	32.00:1	31.37

The fidelity and compression of these (and other) encodings are summarized in table 1. These same results are shown in false color in figures 14, 15, and 16 respectively. The sharp color boundaries of the false color display act to magnify small errors in the encodings. This, combined with the topography of steep cliffs rising directly out of the flat ocean resulted in the distortion along the coastlines, particularly in the ADCT encodings. The compression versus fidelity results indicate that the iterated transform algorithm performed well when compared with the ADCT and MRVQ methods. Other important factors to consider when comparing these compression algorithms are access time, decoding time, and encoding time. Iterated transforms, along with ADCT are variable bit-rate methods, which would result in slower access times than the fixed bit rate MRVQ algorithm. Iterated transform encoding requires an extensive search procedure, making it slower than MRVQ, which also requires a search, albeit a shorter one. Iterated transform encoding is also slower than ADCT, which requires only a transformation and quantization. For most applications, encoding would be a one-time procedure; therefore, encoding time would not necessarily be an important concern. If an application required all or a large fraction of DTED be encoded, then the computer costs for encoding become significant. For many applications, the speed of decoding an image might be a critical requirement. The decoding for iterated transforms is a simple iteration, making it faster than ADCT (where decoding takes as long as encoding), and slower than MRVQ, which is essentially a table lookup.

3. APPLICATION TO COLOR

Application of the iterated transform compression method has, to date, been limited to 6-bpp gray scale images (Jacquin, 1989, 1990) and 8-bpp gray scale images. Another common format for images is 24-bpp color. As equipment that can display these images becomes cheaper and more common, the 24-bpp format will become more widely used. In fact, the Joint Photographic Experts Group (JPEG) image compression standard is being set up for 24-bpp color image. In the 24-bpp format, an image is made up of red, green, and blue subimages, each subimage being represented with 8 bits. Because a relatively large amount of information is used to store each pixel, there is a large potential for compression of these images. This potential for compression is enhanced by the fact that there is often a high degree of correlation between the pixel values of the three subimages. A standard approach for encoding 24-bpp color images is to transform the images from the RGB representation to the YIQ luminance and chrominance representation. The luminance (Y) image (black and white television picture) generally contains the bulk of the information content of the image. The chrominance (I and Q) contains image color content, and generally contains a small fraction of the image information. This allows for spatial averaging of the chrominance with little loss in image fidelity. A flow diagram indicating the encoding and decoding process is given in figure 12.

All the data presented in this section are 512 $\times$ 512 pixel images. The spatial averaging reduced I and Q to 128 $\times$ 128 arrays. During decompression, I and Q

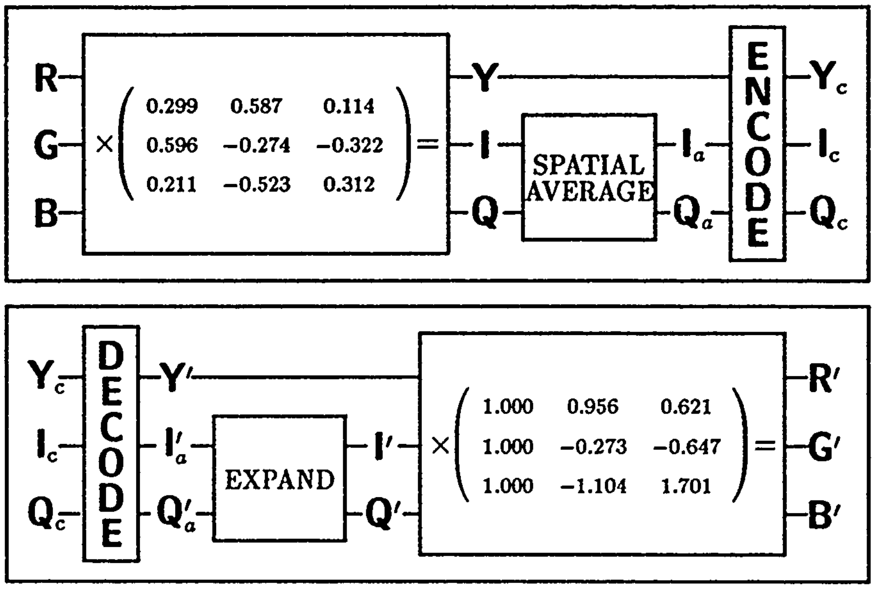

Figure 12 . The encoding and decoding of a color image.

were expanded to 512 × 512 by means of linear interpolation. In figures 17–19 are shown the original images, the iterated transform reconstruction and adaptive discrete cosine transform reconstruction for the image of Lena. Figures 20–22 and 23–25 show the original and both reconstructed images for the vegetables and the mandrill respectively. The fidelity of these images is given by

$$PSNR = -10 * \log_{10}\left\{\frac{\sum_{i=1}^{512}\sum_{j=1}^{512}\left[(R_{ij}-R'_{ij})^2+(G_{ij}-G'_{ij})^2+(B_{ij}-B'_{ij})^2\right]}{(3\times 512^2)\times(2^8-1)^2}\right\}$$

where R_{ij}, G_{ij}, and B_{ij} are the pixel values of the original image and R'_{ij}, G'_{ij}, and B'_{ij} are pixel values of the reconstructed image. The compression and fidelity results for these encodings given in table 2 indicate that the compression versus fidelity performance of the iterated transform method is competitive with, but does not surpass the ADCT results.

A second method for compressing 24-bit images using iterated transforms was also tested. Instead of converting to the YIQ representation, the method takes advantage of the similarity between the R, G, and B images by using the same (or similar) transformations for all three subimages whenever possible. For this reason, the method will be called *common transformations.* With this method, the first step is to encode one color subimage (say the R subimage) in the normal fashion. This will be referred to as a primary encoding. The second step is to perform a secondary encoding of the other images.

Table 2 . Compression and fidelity results for 24-bit images encoded using YIQ approach.

Image	Method	Compression	PSNR(dB)
Lena	Iterated Transforms	65.54:1	29.00
Lena	ADCT	63.44:1	30.15
Vegetables	Iterated Transforms	69.63:1	29.23
Vegetables	Iterated Transforms	90.01:1	28.40
Vegetables	ADCT	60.16:1	30.25
Vegetables	ADCT	73.14:1	29.58
Mandrill	Iterated Transforms	50.95:1	21.38
Mandrill	ADCT	47.20:1	22.25

A secondary encoding is performed by encoding (say the G subimage) using the same set of R_i's as was used to encode the R subimage. Each R_i in the G subimage is encoded using the same w_i (i.e., the same D_i, s_i, and o_i) as was used to encode the corresponding R_i in the R subimage. If the predetermined error criteria is not met for a given R_i, the optimal offset ($o_i{}^{\mathrm{G}}$) is calculated, and replaces $o_i{}^{\mathrm{R}}$, while D_i and s_i remain unchanged. The error is recalculated using the new offset, and compared with the predetermined error criteria. If the error criteria is still not met the optimal offset ($s_i{}^{\mathrm{G}}$) and scale ($o_i{}^{\mathrm{G}}$) factors are calculated to replace $s_i{}^{\mathrm{R}}$ and $o_i{}^{\mathrm{R}}$, with only D_i remaining unchanged. The error is again recalculated using the new scale and offset, and compared with the predetermined error criteria. If the error criteria is still not met, a search is performed for the optimal w_i. To store each $w_i{}^{\mathrm{G}}$, an identification code indicating if $w_i{}^{\mathrm{G}}$ is identical to $w_i{}^{\mathrm{R}}$, $w_i{}^{\mathrm{G}}$ has a new offset, $w_i{}^{\mathrm{G}}$ has a new scale and offset, or $w_i{}^{\mathrm{G}}$ is completely different from $w_i{}^{\mathrm{R}}$, is required, followed by the information not contained in $w_i{}^{\mathrm{R}}$. Scale factors are stored with 5 bits, offsets with 7 bits, and complete transformations with approximately 30 bits. The identification code takes 2 bits to store, so the net savings per transform is 28, 21, 16, or roughly break even (a variable length code, sometimes less than two bits, sometimes more than two bits, is normally needed to identify the range size, and is not required here) depending on whether or not offset, scale, and domain information is required. The last subimage (in this case, the B subimage) is then encoded using the w_i's from the R subimage, just as was done for the G subimage.

This technique can be carried one step further when encoding the last subimage by first checking for common transformations with the primary image, and then checking for common transformations with the secondary image. This additional step requires

Table 3 . Compression and fidelity results for 24-bit images encoded using common transformation approach.

Image	Primary	Secondary	Tertiary	Compression	PSNR(dB)
Lena	R	G & B		62.3:1	27.7
Lena	R	G	B	63.7:1	27.6
Lena	R	B	G	62.8:1	27.5
Lena	G	R & B		57.5:1	28.8
Lena	G	R	B	56.9:1	28.8
Lena	G	B	R	56.9:1	28.8
Lena	B	R & G		64.6:1	27.1
Lena	B	R	G	63.6:1	27.1
Lena	B	G	R	64.6:1	27.1
Vegetables	R	G & B		57.2:1	28.1
Mandrill	R	G & B		21.9:1	21.7

an additional bit for each identification code for the tertiary subimage. Data using the common transformation method on three different images are given in table 3. In the table, results for applying the method with R, G, and B permuted are given for the image of Lena. It is seen that the different permutations of R, G, and B yielded similar results. Permuting R, G, and B in the algorithm when encoding the other two images resulted in smaller variations than for the image of Lena. The results also indicated that the extra bit for the identification code required for the tertiary encoding is sufficient to cancel out any gain in performance that might be attained by using the tertiary encoding. Comparing the results with the data given for the YIQ method shows that the YIQ method is a superior method. This was especially true in the case of the image of the mandrill, where the great bulk of the information is in the Y subimage, and the correlation between the R, G, and B subimages is relatively small.

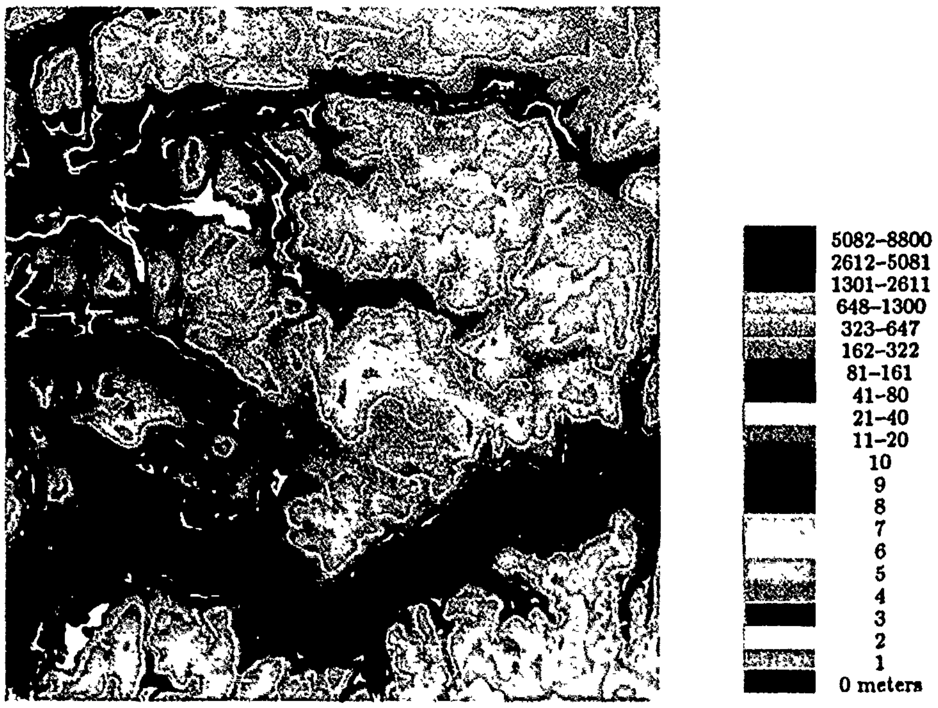

Figure 13 . The false color version of figure 6, and the elevation legend for false color images.

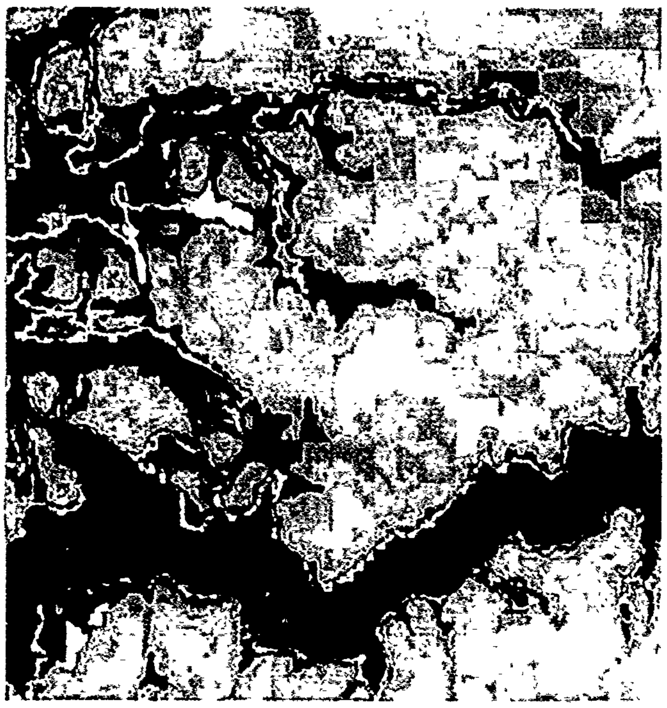

Figure 14 . The false color decoded iterated transform Norway image.

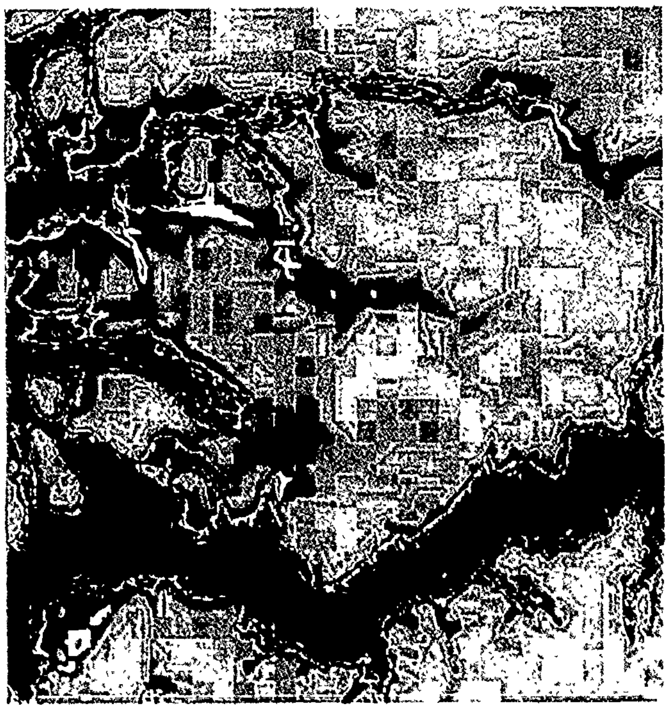

Figure 15 . The false color decoded ADCT Norway image.

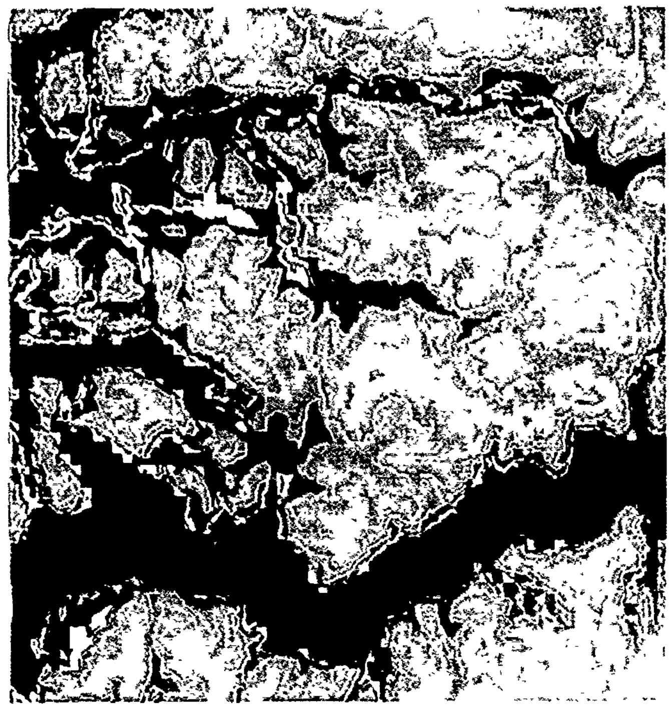

Figure 16 . The false color decoded MRVQ Norway image.

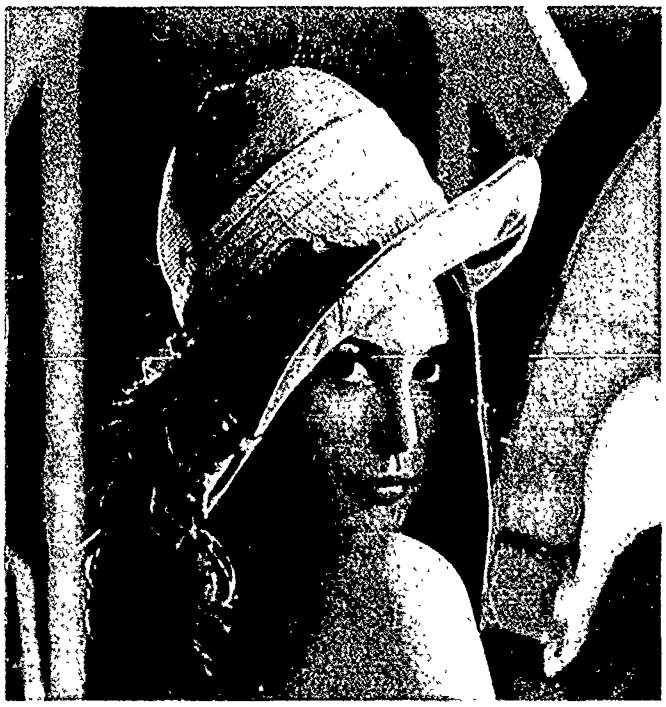

Figure 17 . The original color image of Lena.

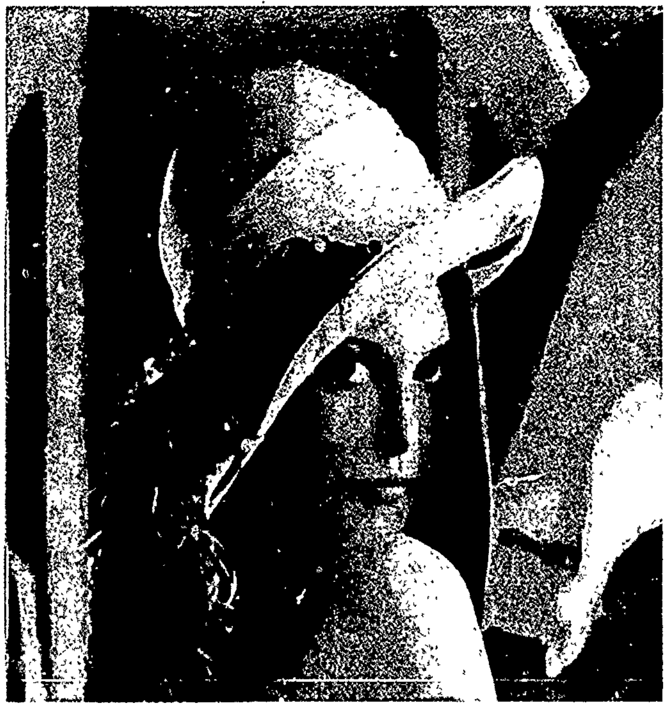
Figure 18 . The decoded iterated transform color image of Lena.

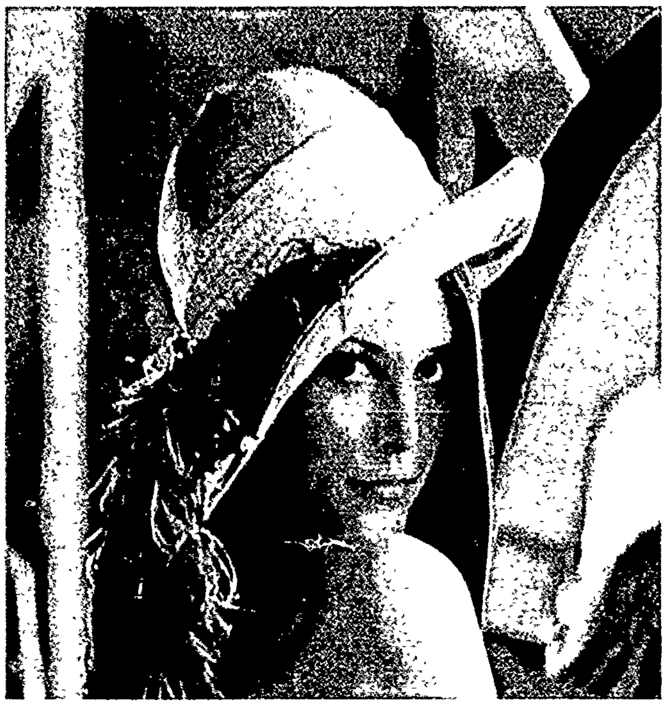
Figure 19 . The decoded ADCT color image of Lena.

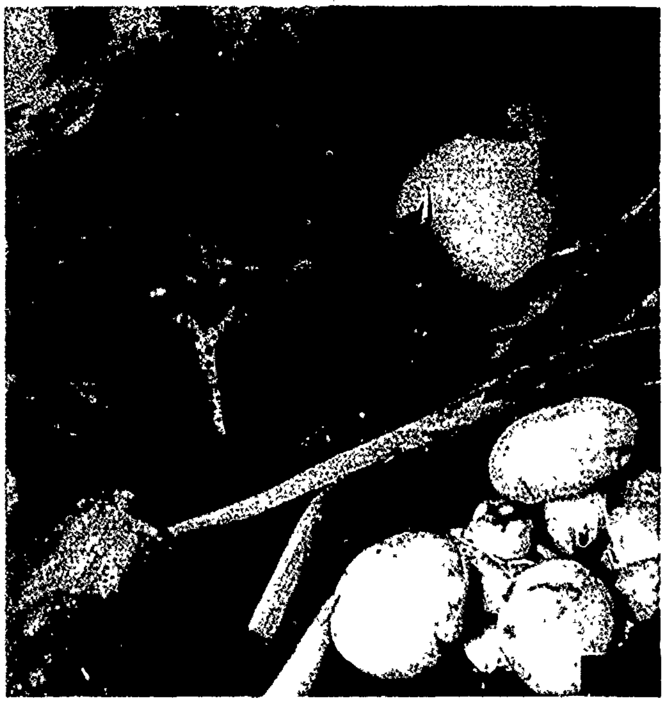

Figure 20 . The original color image of vegetables.

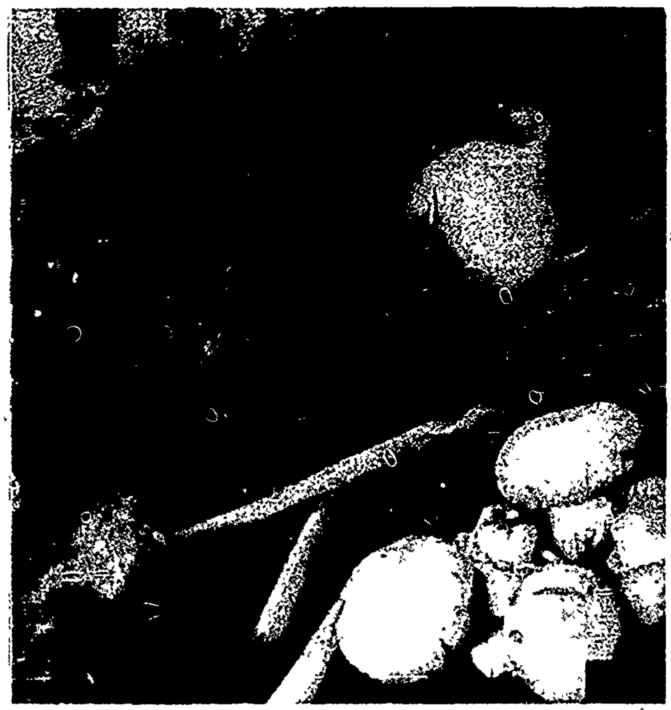
Figure 21 . The decoded iterated transform color image of vegetables.

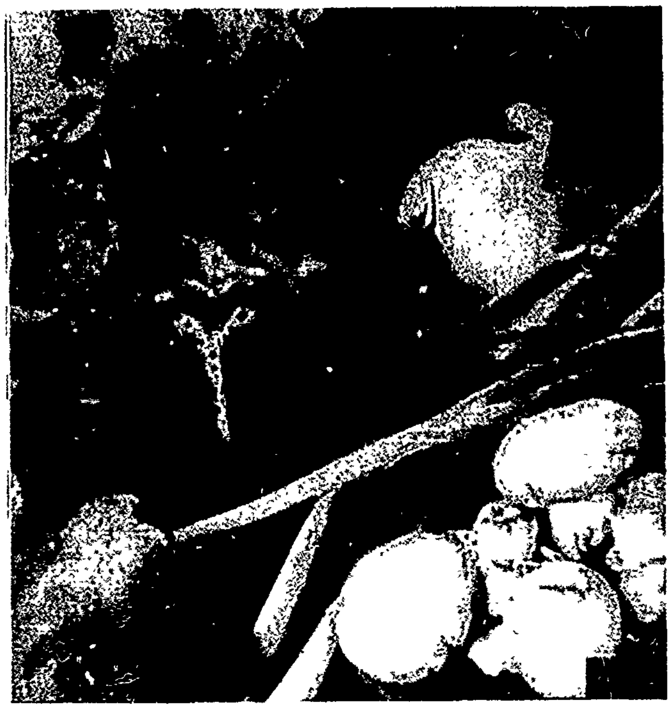
Figure 22 . The decoded ADCT color image of vegetables.

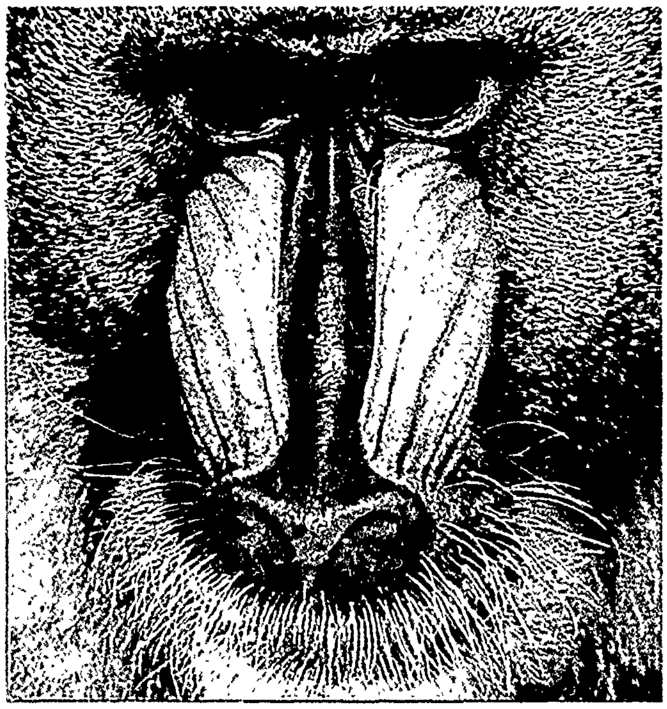

Figure 23 . The original color image of the mandrill.

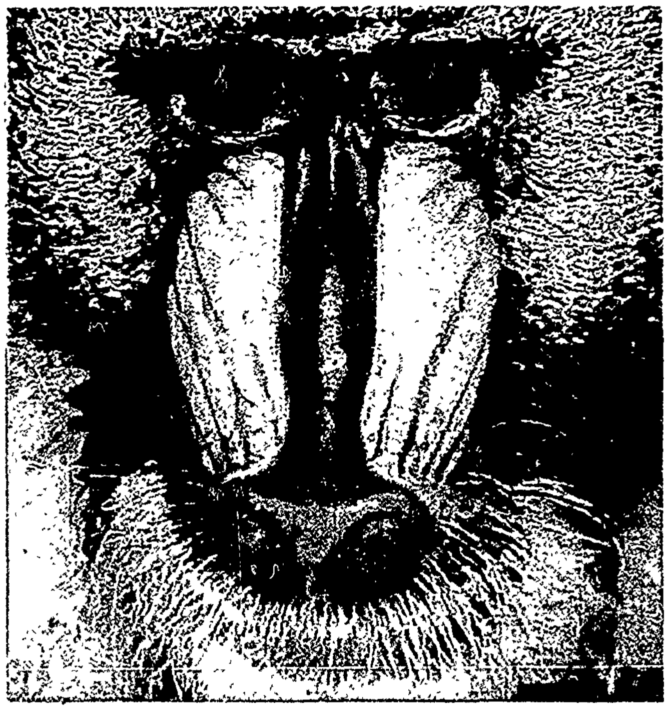

Figure 24 . The decoded iterated transform color image of the mandrill.

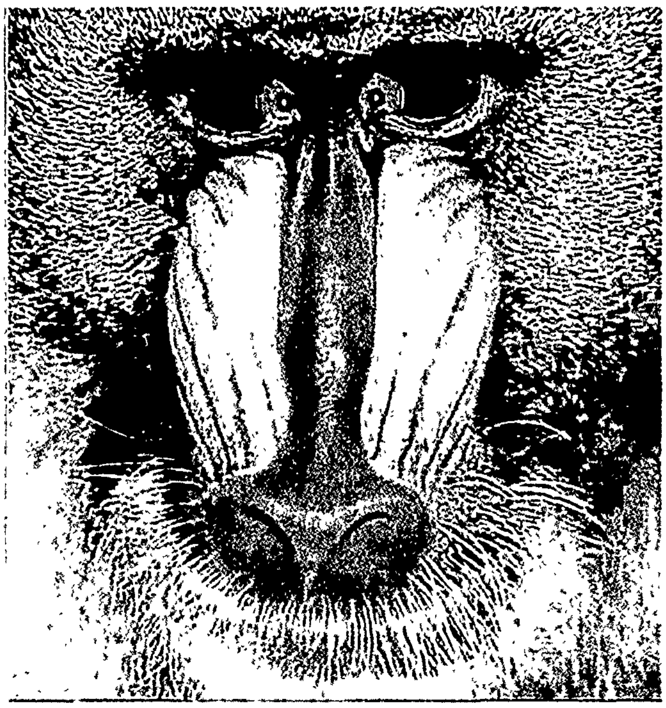

Figure 25 . The decoded ADCT color image of the mandrill.

4. REFERENCES

Alward, H.L. and D.A. Nicholls. 1986. "Hierarchical Data Structures for a Digital Terrain Map System," AFWAL-TR-86-1177, available from DTIC.

Barnsley, M.F. 1988. *Fractals Everywhere*, Academic Press, Inc., San Diego, CA.

Barnsley, M.F. and A.E. Jacquin. 1988. "Application of Recurrent Iterated Function Systems to Images," *SPIE vol. 1001, Visual Comm. and Image Processing*, p. 122.

Barnsley, M. F. and A.D. Sloan. 1988. "A Better Way to Compress Images," *Byte*, vol. 13, p. 215.

Chen, W. and W.K. Pratt. 1984. "Scene Adaptive Coder," *IEEE Transactions on Communications*, vol. 32, pp. 225–232.

Fisher, Y., E.W. Jacobs, and R.D. Boss. 1991. "Iterated Transform Image Compression," NOSC TR 1408, Naval Ocean Systems Center, San Diego, CA.

Jacobs, E.W., Y. Fisher, R.D. Boss. 1991. "Image Compression: A Study of the Iterated Transform Method," *Submitted to Signal Processing.*

Jacquin, A.E. 1989. "*A Fractal Theory of Iterated Markov Operators, with Applications to Digital Image Coding*," Ph.D. Thesis, Department of Mathematics, Georgia Institute of Technology.

Jacquin, A.E. 1990. "A Novel Fractal Block-Coding Technique for Digital Images," *IEEE ICASSP*, vol. 4, pp. 2225–2228.

Jacquin, A.E. 1990. "Fractal Image Coding Based on a Theory of Iterated Contractive Image Transformations," *SPIE Visual Comm. and Image Processing '90*, vol. 1360, pp. 227–239.

Linde, Y., A. Buzo, R.M. Gray. 1980. "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Comm.*, vol. COM-28, no. 1, pp. 84–95.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

ONLY *(Leave blank)*	2. REPORT DATE December 1991	3. REPORT TYPE AND DATES COVERED Final: Oct 1990 — Sep 1991
...E AND SUBTITLE STUDIES OF ITERATED TRANSFORM IMAGE COMPRESSION AND ITS APPLICATION TO COLOR AND DTED		5. FUNDING NUMBERS PE: 0602936N PROJ: RV36I21 SUBPROJ: 63–ZE88–01 ACC: IC000037
6. AUTHOR(S) R. D. Boss and E. W. Jacobs		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Ocean Systems Center San Diego, CA 92152–5000		8. PERFORMING ORGANIZATION REPORT NUMBER NOSC TR 1468
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Ocean Systems Center San Diego, CA 92152–5000		10. SPONSORING/MONITORING AGENCY REPORT NUMBER In-house
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A new classification technique based on *archetypes* can reduce encoding time and still maintain compression and fidelity. The use of different metrics has shown little differences in the final encoding quality thus implying that the use of the root mean square metric is preferred.

Also presented are Digital Terrain Elevation Database (DTED) that are more accurate after using an iterated transform algorithm modified to encode the coastlines. The results compare favorably with an adaptive discrete cosine transformation (ADCT) and mean residual vector quantization algorithm. Additionally, results of two iterated transform algorithms for encoding 24-bit/pixel color images are presented. The first method transforms the red, green, blue (RGB) to the luminance-chrominance representation before encoding, showing results similar to ADCT algorithms. The second method encodes RGB subimages directly using common transformations, and does not perform as consistently as the luminance-chrominance method.

14. SUBJECT TERMS Digital Terrain Elevation Database (DTED) iterated transforms	archetypes fractals	image compression	15. NUMBER OF PAGES 40 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18 SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAME AS REPORT

UNCLASSIFIED

21a. NAME OF RESPONSIBLE INDIVIDUAL Bill Jacobs	21b. TELEPHONE *(include Area Code)* (619) 553–1614

NSN 7540-01-280-5500

Standard form 298

UNCLASSIFIED

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 014	W. T. Rasmussen	(1)
Code 0141	A. Gordon	(1)
Code 0142	K. J. Campbell	(1)
Code 0144	R. November	(1)
Code 414	D. Gomez	(1)
Code 414	F. Martin	(1)
Code 50	H. O. Porter	(1)
Code 57	R. H. Moore	(1)
Code 573	J. C. Hicks	(1)
Code 573	E. W. Jacobs	(40)
Code 60	F. E. Gordon	(1)
Code 952B	J. Puleo	(1)
Code 961	Archive/Stock	(6)
Code 964B	Library	(3)

Defense Technical Information Center
Alexandria, VA 22304-6145 (4)

NCCOSC Washington Liaison Office
Washington, DC 20363-5100

Center for Naval Analyses
Alexandria, VA 22302-0268

Navy Acquisition, Research & Development
Information Center (NARDIC)
Alexandria, VA 22333

Navy Acquisition, Research & Development
Information Center (NARDIC)
Pasadena, CA 91106-3955